Toward Adaptive Dual Expert and Intelligent Tutoring Systems in Medicine: A Case Study for Spinal Injuries Diagnosis

John Halloran and Theodoros N. Arvanitis

CSRP Number 466

June 25, 1997

ISSN 1350-3162

UNIVERSITY OF

SUSSEX

AT BRIGHTON

Cognitive Science

Research Papers

Toward Adaptive Dual Expert and Intelligent Tutoring Systems in Medicine: A Case Study for Spinal Injuries Diagnosis

John Halloran and Theodoros N. Arvanitis

School of Cognitive & Computing Sciences
University of Sussex, Falmer, Brighton, BN1 9QH, UK
Email: {johnhall}{theoa}@cogs.susx.ac.uk

June 25, 1997

Abstract

Traditionally, a classical AI model of intelligence has informed the design of Expert Systems and Intelligent Tutoring Systems for medical applications. This paper[1] argues that such a model of intelligence produces educationally atypical results in that it intrinsically reduces the capacity of such systems to deliver an educational product – the effective learning of a medical domain – with human-like efficiency. Systems based on an adaptive model of intelligence, one which emphasizes intelligence as **adaptation to dynamic environments** and as **distributed**, can help (a) transform Intelligent Systems into more human environments, and (b) establish **educational robustness** – the reliability, durability, and utility of educational software in a given educational sphere. The paper provides evidence for this claim through the discussion of a case study: an adaptive dual system for teaching diagnosis of spinal injuries. The system contributes to the establishment of general principles for the design of adaptive educational software for medical diagnosis. Central to these are that an adaptive system functions as a cognitive model; and that the concept of the **Intelligent Agent** is likely to be importantly implicated.

Key concepts: software adaptivity, knowledge environment, user environment, educational typicality, educational robustness, Intelligent Agent, dual system.

[1] A revised version of this paper, entitled "Toward dual adaptive expert and intelligent tutoring systems: case study in medical AI" appears at the Proceedings of the International Conference on Cognitive Systems, 1996.

PART ONE: THEORETICAL FOUNDATIONS

1 Introduction: educational robustness and software adaptivity

The system discussed in this paper is a piece of educational software designed to teach GPs and medical students to successfully identify the symptoms of spinal injuries, make correct diagnoses, and suggest appropriate treatments; as well as performing these functions automatically. The major motivating issue is whether *educational robustness* – the ability of an educational service provider to reliably and repeatedly deliver an educational product, here medical training – can be achieved, where the provider is a piece of software, through *software adaptivity*; that is, the ability of software to fit itself to individual user needs and characteristics, as well as to a dynamically changing knowledge base. The assumption is that educational robustness necessarily involves adaptivity, since educational robust human teaching routinely involves addressing individual learning needs as well as continuous retraining in the knowledge domain.

2 Theoretical principles of software adaptivity

2.1 Cognitive model of diagnosis

How do doctors actually make diagnoses? Literature suggests that this is a process of *differential*. For a given set of presenting symptoms, there may be more than one consistent diagnosis. This diagnostic under-determination can be reduced with an increase in data. However, it is often the case that data is ambiguous or incomplete; or that the data is unique. In such cases, diagnosis can be made through analogy with other known cases which more or less nearly conform to the current data [12]. This means that diagnosis is both data-driven and analogy-based. Analogy-making appears to be based on implicit rules [9]. A system which automates such a process, then, should be a cognitive model of data-driven analogy-making. Such a model would elucidate the cognitive processes involved in expert diagnosis as well as assisting educational robustness in that the model would accurately represent the human diagnostic methodology, with the implication that it is humanly learnable.

2.2 Classical AI theory of intelligence

Educational software involves systems regarded as intelligent. Therefore, such systems are informed by a theory of intelligence. Marr [10] discusses three levels in theory-building: the *computational* level, concerned with the purposes of the theory; the level of the *algorithm*, at which the theory is operationalised; and the *hardware*

level, which specifies the physical base that instantiates the algorithm. In terms of theories of intelligence, cognitive modelling in classical AI has tended to neglect Marr's "hardware" level, since it has often been argued that the psychological level of analysis is irreducible to the physical [6]. This has tended to mean that cognitive models of intelligence are algorithmic theories, not hardware theories. Traditionally, the model dominating theories of intelligence has been the physical symbol system hypothesis, [4], [5]. Fodor's Language of Thought Hypothesis (LOT) is a paradigm example: it claims that there are cognitive tokens in the mind which carry meaning, physically instantiated, which combine in rule-governed, law-like ways. In other words, intelligence is seen as arising out of the operation of rules over a database of static information atoms.

There are three major drawbacks with the application of a symbol system theory of intelligence to educational software in medicine; that is, Expert Systems (ESs) and Intelligent Tutoring Systems (ITSs). First, schematic ossification. According to schema theory [11], [2], [1], schemas are flexible information packs that develop dynamically in relation to new experience. This means that an ITS built on such a theory is unlikely to make a flexible teacher. Second, explicit rule operation over a static database is not a good model of analogy-building on the basis of implicit rules and a dynamic knowledge base, which means that an ES built on such theory would not routinely feature such capabilities. Third, the downgrading of the importance of a hardware level of theorising means that the theory does not provide hardware principles in the design of educational software: principles which might usefully constrain, or even define, algorithms.

2.3 Adaptive theory of intelligence

For Michael Wheeler [13], intelligence is a function of adaptive potential: "On evolutionary grounds, it seems reasonable to suppose that human linguistic competence and deliberate though are overlays on a prior ... capacity for adaptive behaviour". Wheeler argues that "we should identify a system as an adaptive system only in those cases where it is useful to attribute survival-based purpose ... to that system" [13]. Clearly, many computer systems are not evolved. Therefore, given this constraint, adaptiveness is redefined as "a matter of surviving long enough in an environment to achieve certain goals" [13]. This view of intelligence, then, implies that any intelligent system must first be adaptive: it should "survive in an environment long enough to achieve certain goals". This suggests that one test of adaptivity is that software is survivable; that educational software is educationally robust enough to remain in use.

Added to this, the dynamical systems view of intelligence regards it as *distributed*. Hutchins [7], [8] argues that the accomplishment of a cognitive task involves the interaction of agents. This suggests that the value of a piece of educational software lies in its contribution to an educational outcome also involving

the user. This means it should interface to the user in such a way that the task is achievable. This implies social and communicative abilities.

Intelligence as adaptation points up the fact that software exists in an environment - in fact, multiple environments: an environment of knowledge; and an environment of users. To be survivable, the implication of this is that software develop in tandem with those environments, responding to dynamic change. This immediately raises the question, again, of whether a flexible system inhabiting dynamic environments can be achieved if it consists of a fixed algorithm (rule-set) operating over a fixed set of symbols, as on the classical AI theory of intelligence.

2.4 Adaptation to a knowledge environment

The assumption here is that, like the environment of an organism, the knowledge domain inhabited by a piece of software is dynamic. This seems highly plausible given volatile domains where there is a knowledge turnover with new data-collection methods and an expanding and changing cases corpus. Hence, adaptive software should readily accommodate change in its knowledge base. This would contribute to educational robustness in that curriculum was continually and relevantly updated.

2.5 Adaptation of user environments

Each user is conceived as a unique environment and the adaptation is that the software should be capable of addressing individual users on the basis of their needs. This should contribute to educational robustness in that a system is optimally 'tuned' to the specific users it starves. Adaptation to user environments would help address two problematic issues with educational software:

1. *System load.* Educational software requires that users be able to operate it in order to learn from it: this is frequently not addressed, which increases the cognitive load incurred by a user. Hence, software should act as a *system tutor*, interpreting the user in terms of expertise in operating the system, and offering relevant guidance. In addition, a system should feature a cognitively transparent interface: one whose functions are easy to understand and operate.

2. *Educational load.* ITSs have been criticised in that they often feature restricted and schematic ways of representing users, which means that users are forced into particular learning styles, increasing cognitive load. An adaptive system should reverse the need for the user to adapt to the system by featuring an architecture which allows the system to adapt to the users.

2.6 Architecture of software adaptivity

Intelligent Agents: Intelligent Agents (IAs) are software entities which have been recognised as importantly implicated in software adaptivity. They have been defined in terms of four criteria [14]:

1. *Autonomy*
   This means the Agent is self-monitoring; it has self-created control; it acts on behalf of the user without any human intervention.

2. *Social Ability*
   Agents are capable of interaction with other Agents and with humans.

3. *Reactivity*
   Agents perceive an environment and react to it "in a timely fashion"[14].

4. *Pro-activity*
   Agents are not just reactive to an environment. They can "exhibit goal-directed behaviour by *taking the initiative*" [14]. This implies that they can *change* an environment.

These criteria have a strongly adaptive favour. The criteria of reactivity and pro-activity are concerned with the fitness within environments; autonomy suggests that software be able to self-develop in response to environmental situatedness; and sociality implies the communicative potential discussed above. This suggests that we can use Intelligent Agents in designing adaptive software. However, the definition is functional not operational; there is no given hardware or algorithm specification. Thus, it may be that where software satisfies these functional criteria, the results will be useful as algorithm and hardware theories of intelligence.

Neural implementation of Expert Systems: Neural nets are intrinsically non-symbolic and so there is no cross-referencing or other issue in updating a dynamic knowledge base. At the same time, neural nets are automatic classifiers, both in the sense that patterns can be coupled to targets in a highly robust way and that any pattern within a certain tolerance of the original will retrieve the target; and in that nets build implicit schemas which can be used to access analogies, where analogies are similar patterns. At the same time neural nets are data-storage engines. This suggests that on model of data-driven and analogy informed diagnosis on the basis of 'intuitive' or implicit rules discussed above, neural architectures might make good models of intelligence in that they would, in principle, feature the adaptive potentials required. Since neural nets are hardware hypotheses of intelligence inspired by the structure of the brain, their usefulness as adaptive ESs might suggest that adaptive software is neurally implemented and the hardware level of theorising.

Neural Intelligent Tutoring Systems: A common criticism of symbolic ITSs is that they are over-schematic and restrictive in the way they assess users [3]. Users are interpreted according to fixed system schemata. This means that ITSs are often schematically ossified, placing both system load and educational load on users. While educational load is not educationally atypical, good teaching ought to routinely reduce it; but such teaching is adaptive and based on flexible schemas. However, system load is educationally atypical, and therefore should not feature if a system is to be educationally robust. We have seen that neural nets are spontaneous schema builders. This means that user schemas could be built through neural implementation of ITSs. Such schemas would be adaptive to users, so might remove the problems of schematic ossification and system load associated with non-adaptivity.

Dual Systems: Functional discreteness of the ES and ITS modules of a piece of educational software would increase the survivability and educational robustness of the software in that the system could be used by users at any level of expertise. Non-experts could use the ITS with the ES as an embedded component, while experts would be able to make use of the decoupled ES.

PART TWO: METHOD

The system, which will be referred to as ADS - Adaptive Dual System -, is housed by a graphical user interface (GUI) in which an expert system and an Intelligent Tutoring System are functionally discrete graphical objects. The ADS is designed to make diagnoses on the basis of data and analogy, and to suggest treatments given a case corpus. At the same time, it teaches the principles involved in deriving and interpreting data, and in suggesting appropriate treatments.

3 The Expert System

The Expert System is designed to store correlations between sets of symptoms ('symptomsets') and diagnoses; and to retrieve an optimal treatment given a diagnosis.

3.1 Symptomset-to-diagnosis knowledge engineering

The first set of correlations is achieved in the following way. Symptomsets are defined as the presence or absence of a symptom over a complete set of known variables. This binary representation means that the resulting pattern can be input to a three-layer backpropagating neural network. This net trains symptomsets to diagnoses, represented as arbitrary binary patterns. To retrieve a diagnosis, the users

input symptoms represented as pushbuttons; if a button is pressed the symptom is present. Blocks of related symptoms (for example, related to resisted movement or passive movement) are separated and represented as different graphical objects for cognitive transparency. The user can input new symptomset-to-diagnosis relationships by using the same process. The user inputs the diagnosis in natural language and this is automatically translated into a binary representation. This means that these correlations are easily updatable, and that the system is equipped with the means of dynamically updating its knowledge base.

3.2 Neural classification

Neural nets spontaneously group similar patterns. This has two implications. First, partial patterns that are similar to stored patterns will retrieve the target for the stored pattern at some level of tolerance. Second, highly different patterns will not interfere. In this way, patterns can be more or less proximal, which has implications for analogical transfer.

3.3 Neural analogical transfer

Problem-solving has been recognised to involve *analogical transfer* [9]. Analogical transfer means that a new problem is solved in terms of an existing analogous problem. For diagnosis, this means that a novel symptomset might be interpreted in terms of the nearest analogy. Because neural nets group similar patterns as schemas, this means that the nearest analogy – a stored pattern – will automatically be retrieved. This means that the content addressability of neural nets makes them intrinsically effective analogy-makers. Analogical transfer where there is more than one analogy can also be modelled. This is because of the ability of nets to train correlations at different levels of salience. For example, if correlation $a-b$ is trained five times (that is, there are five occurrences of the correlation in the training set), while the correlation $a^2 - b$ is trained twice, this means that pattern a, since it is more highly trained than a^2, will be more salient as a trigger for target b. This implies that analogy-making is done on the basis of data but also on statistical weighting: the chosen analogy is most heavily represented in terms of the current data.

The system, then, makes analogies in the case of unknown symptomsets. The system provides a natural language record of the selected diagnosis to one of the GUI text-display modules, and also informs the user of the difference between the net output and the diagnosis selected on analogy or directly. Tolerance is interpreted as an index of soundness of the analogy.

3.4 Diagnosis-to-treatment knowledge engineering

Once a diagnosis is retrieved by the ES, the user has the option of instructing the system an appropriate treatment.

3.5 Many-to-one training restriction by backpropagation

Standard backpropagation architectures limit training to many-to-one. What this means is that the number of different patterns can be trained to a single target, but not vice versa: it is not possible to train one pattern to different targets. This means that the advantages of neural nets – differential salience through weighting, schema-building, analogical transfer – are lost when we wish to choose between competing treatments. Often in differential diagnosis there is no way to enrich the data and reduce the under-determination. If more than one treatment is available, we need to know which one to administer. Here, case-based reasoning can be used. This simply means, choose the treatment that has been most successful given the same diagnosis; or given an analogous diagnosis. However, if the same diagnosis cannot be trained to different targets, this means that we cannot neurally represent such case-based reasoning. To fall back on a symbolic paradigm is an unsatisfactory solution since we risk losing the adaptive advantages of neural implementation.

3.6 The many-to-many training protocol and neurally implemented case-based reasoning

This problem has been addressed through an original 'many-to-many' training protocol developed as part of the research. This protocol simply involves representing a correlated diagnosis and a target as a single 'spliced' pattern where the diagnosis is the first section and the treatment is the second. The target for the pattern is the pattern itself. Salience levels are retrieved through presenting the retrieved diagnosis together with the numerical average of all correlated treatments. This results in the derivation of the most salient treatment. Thus, case-based reasoning, that is, the selection of the most statistically salient treatment, is achieved automatically through the architecture of the neural net.

3.7 Adaptation to dynamically changing knowledge environment

As the system generates new diagnoses and treatments of these, it adds these cases to the case corpus it access in order to select treatment. Periodically, when the information becomes available, the user inputs a success rate for the treatment. At this point, the net prepares to retain itself: the rate is compared with others for the same diagnosis-treatment correlation and the result, a general success rate is

used to decide the number of presentations of the correlation in the new training set. This is one form in which the system creates and responds to a dynamically changing knowledge environment.

4 The Intelligent Tutoring System

The ITS is designed to teach the principles of diagnosis for the domain of spinal injuries. It does not explicitly teach how treatment should be recommended since it is based on accessing a case corpus which develops as a function of user's medical experience. The ITS is built around two main engines: first, a symbolic query- and level-switching architecture that generates implicit student models and dynamic curricula; and second, a schema-building neural architecture.

4.1 Behavioural/Cognitive teaching methodologies and user level

The ITS features two teaching levels. This first is based on a behaviourist methodology. This teaches the ability to group complete symptomsets for commonly occurring cases and link these to diagnoses. This is entirely text-based. The second features a cognitivist methodology designed to teach users how to derive symptomsets independently and to recognise the underlying reasons for relationships of symptoms. This is done by visual display of animations representing the physical manipulations from which diagnoses are derived. These displays are linked to a symptomset so that, for example, a given manipulation is accompanied by a text message indicating pain; then the next by spasm, and so on. The behavioural level is regarded as a necessary precursor to the cognitive.

4.2 Query-switching

A dynamic user-adaptive curriculum which also acts as an implicit student model can be built through the use of parallel query types over the two levels. At the behavioural level, this query type is a 'Starter' multiple-choice question format. Users are presented with an 'anchor' set of three definite symptoms from the symptomset being taught, printed as their natural language forms to one of four text presentation windows of the relevant GUI ITS module. A set of correlated symptoms is printed to a different, randomly chosen presentation window; and two distractor sets are also displayed in randomised windows. The user chooses the correlated set. If the user is highly successful, the system switches to the second query-type, 'Intermediate' multiple-choice format. This resembles the 'Starter' format but the anchor set is randomised so that the detection of the correlated set is less easy and requires greater familiarity with the symptomset grouping. Success on this query

type means the user is switched to 'Open' query type. Here, the user is required to input all the symptoms for the symptomset by means of the ES pushbutton modules. The system then provides a breakdown of the total symptomset, the number of correctly identified symptoms, and the success rate. If this is high the user is switched to the cognitive level on the same symptomset; if not, the system repeats the same protocol for the remaining symptomsets. At the cognitive level, the visual displays replace the anchor set at the behavioural level; otherwise, the method is the same.

4.3 User schemas

The query-switching method means that a given resource – a query type and a level – is matched to the user at every stage dependent on the level of success of the user on that query type. The methodology means that levels and query types can be dynamically switched. For example, if the user is consistently unsuccessful on the 'Starter' query type at the behavioural level s/he remains at that level; but if successful, s/he is switched up. Equally, progress can be followed by regress. For each symptomset being taught, responses are logged as vectors and saved to a file. For each symptomset, a different vector exists, reflecting that some may be easier to learn than others. These files are records of user responses which are then analysed as a curriculum profile through unsupervised neural learning. This produces a generalised user profile. Next time the user logs in and registers his/her identity, the system invokes a user schema to define the new curriculum. The user schema is de-feasible at any point and is a heuristic for curriculum than definitive of it. From the new run, a new set of vectors is derived, and a new run profile is produced and integrated with the existing profile; and so on. At the point where the profile indicates completely correct responses, the suer is defined as Expert and the ES ceases to be relevant, unless the user wishes to defeat the operation of the schema and re-use the ITS.

PART THREE: DISCUSSION

In Part One several claims were made. First, that doctors make diagnoses on the basis of implicit rules based on data and analogy. This should mean that an ES which automates this process should be a cognitive model. Classical AI systems, it was claimed, are limited in making such a cognitive model since the underlying model of intelligence is not well-suited to modelling implicit rules and analogy-making; and because they often fail to address a hardware level. It was claimed that systems founded on such a view of intelligence are educationally atypical in that they place an abnormal cognitive load on users. This means that the dissection of an intelligent process in classical AI terms has meant that a human environment

has been lost; especially one in which educators routinely adapt to a knowledge domain and to learners to reduce cognitive load. Therefore, more human educational software ought to involve an alternative theory of intelligence. An adaptive theory of intelligence has several important implications. First, that an intelligent system is survivable and adaptive to dynamic environments; that such a system should be more human and educationally typical/robust by virtue of its adaptive potential; and that the implementation of adaptive algorithms as a working model should provide a hardware theory of adaptive intelligence; and that this could be provided through neural implementation. How far do these claims stand up? Is software adaptivity related to more human-like decision-making and educational processes? Does this produce more educationally robust systems? Are these systems neurally implementable; and do they provide a hardware theory of adaptive intelligence which might constrain future algorithmic theorising? What role do Intelligent Agents play in software adaptivity ?

4.4 Assessment of the Expert System

The Expert System reliably retrieves stored targets when stored patterns are input; it also retrieves analogies. When presented with diagnoses the system has been shown to be reasonably robust in retrieving the most successful treatment over a given case corpus. These results suggest that a neurally-implemented Expert System can adapt to a dynamic knowledge environment to produce reasoning similar to that practiced by doctors.

The neural implementation of case-based reasoning, achieved through the many-to-many training protocol, does involve some problems. At present, averaging of all treatment representations for a given diagnosis can produce anomalous results by virtue of the way treatments are represented; as vectors of four numbers only. Treatment representations are insufficiently differentiated using such a compressed representation, which can mean that the resulting average retrieves inappropriate treatments. Currently this problem is overcome by using an anomaly-checker; this, however, is an inelegant add-on. More extended representations should enable the required differentiation and the removal of this checker. However, despite this limitation, the suggestion is that an ES for medical diagnosis can be achieved neurally, and this lends weight to the claim that adaptive intelligence might, at hardware level, be implemented neurally, since the result is more human than a symbolic counterpart. The fact that such systems might produce human-like reasoning suggests a capacity of conformance with human medical decisions. However, such tests have not yet been run; therefore this is conjecture.

4.5 Assessment of the Intelligent Tutoring System

The ITS schema-building capability means that each user is directly modelled. In this sense, the system adapts to unique user environments. This helps address the issues of schematic ossification and reduction of cognitive load. However, adaptive potential is restricted in that the system simply applies the same methodology in different ways. This raises the whole question of how systems can be built which spontaneously generate new methods in the light of 'teaching experience'. A short-term solution is to expand the range of available methodologies. This raises the whole question of how *autonomy* can be built into adaptive systems: how systems can spontaneously develop new educational methods. At the same time, the system is only adaptive in a theoretical sense since no large-scale user evaluation has been carried out to see how far users are successful in learning by means of the system, so that actual robustness and survivability are only theoretical. However, despite this, the system does build implicit student models and lay out curriculum appropriately given a specific user, and despite the lack of comprehensive testing, the fact that the system works outside a classical paradigm which insists on explicit student modelling and level allocation does not appear to demonstrate that adaptive intelligence in ITSs can be modelled; and that at a hardware level the implementation is at least partly neural.

4.6 Intelligent Agents

In Part One it was pointed out that Intelligent Agents are attempts to implement adaptive intelligence computationally, and that Agents should feature four criteria of autonomy, sociality, reactivity and pro-activity. However, architecture for Intelligent Agents has not been specified; at the same time it is unclear at what level an Intelligent Agent should be characterised. Is an Agent the entire piece of software, for example, or subroutines? It seems most useful for our purposes to conceptualise an IA as the total software, although further research needs to be conducted so that this thesis could be implemented in practice. The results suggested that reactivity and pro-activity can be modelled neurally. The issue of sociality is ambiguous in the context of distributed cognition, since any piece of software could be seen as social if it communicates with a user. This can be achieved through transparent interfaces which reduce system load and constrain interactions in sensible ways. Currently the system tutor is simply a help system which does not differentiate between users; and so this could not be conceived as an adaptive capability of the system. However, such a system ought to involve that same schema-building and implementation processes used by the ITS. The result would be more or less fine-grained help systems. Perhaps the main issue for the system developed here in terms of its status as an Intelligent Agent is whether it is really adaptive in the sense that it can take independent decisions about the form it takes: whether it

can become autonomous. We saw that the ITS was limited to the set of algorithms it can carry out; there is no way for the system to increase its adaptivity through spontaneous redevelopment. This means that a teleological view must logically be taken: one which anticipates various learning styles of the users it serves and provides algorithms to meet these needs.

5 Conclusions

The system discussed here attempts to operationalise an adaptive theory of intelligence to produce adaptive software that aims educational robustness, which implies human-like and human-friendly characteristics. It was hoped that the implementation of human-like characteristics in both expert reasoning and tutoring might logically suggest a hardware theory of adaptive intelligence. It was found that it is possible to implement a system which is adaptive to an extend by using non-symbolic architectures. This suggests that adaptive intelligence does not necessarily use or need explicit rules or complete information. The relationship of software adaptivity to educational robustness has only been demonstrated in principle and requires user evaluation; at the same time the system is constrained by its omissions (for example the system tutor) and certain outstanding problems. The most significant of these is the resolution of the many-to-many training problem to remove anomalies. This in itself is an interesting finding in that it suggests that trainable neural nets for cognitive modelling are limited in a psychologically implausible way; and that more substantial work on the many-to-many training problem needs to take place.

References

[1] A. D. Baddeley, *Human Memory: Theory and Practice*, Lawrence Erlbaum, 1990.

[2] G. Cohen, G. Kiss, and M. LeVoi, *Memory: Current Issues*, 2nd edition, Open University Press, 1993.

[3] S. J. Derry, and S. P. Lajoie, "A middle camp for (un)intelligent instructional computing: an introduction" in *Computers as Cognitive Tools*, S. J. Derry and S. P. Lajoie (eds), Lawrence Erlbaum, 1993.

[4] J. A. Fodor, *The Language of Thought*, Thomas Crowell, 1975.

[5] J. A. Fodor, *Psychosemantics*, Bradford Books/MIT PRess.

[6] J. A. Fodor, and P. Pylyshyn, "Connectionism and cognitive architecture: a critical analysis", *Cognition*, 1988;28:3-71.

[7] E. Hutchins, "The social organization of distributed cognition" in *Perspectives on Socially Shared Cognition*, L. B. Resnick, J. M. Levine, and S. D. Teasley (eds), American Psychological Association, 1993.

[8] E. Hutchins, *Cognition in the Wild*, Bradford Books/MIT Press, 1995.

[9] H. Kahney, *Problem Solving*, Open University Press, 1986.

[10] D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W. H. Freeman, 1982.

[11] J. Piaget, *The Child and Reality: Problems of Genetic Psychology*, Grossman, 1973.

[12] K. F. Schaffner, *Logic of Discovery and Diagnosis in Medicine*, University California Press, 1985.

[13] M. Wheeler, "From robots to Rothko: the bringing forth of worlds" in *The Philosophy of Artificial Life*, M. Boden (ed), OUP, 1995.

[14] M. J. Wooldridge, and N. R. Jennings, "Intelligent Agents: Theory and Practice", submitted to *Knowledge Engineering Review*, 1994, Revised, 1995.